

APPLICATION
FOR
UNITED STATES LETTERS PATENT

TITLE: UNIFIED DESIGN PARAMETER DEPENDENCY
MANAGEMENT METHOD AND APPARATUS

APPLICANTS: WILLIAM R. WHEELER, MATTHEW J. ADILETTA
AND TIMOTHY J. FENNELL

CERTIFICATE OF MAILING BY EXPRESS MAIL

Express Mail Label No. EL584B11313US

Date of Deposit 11-26-01

I hereby certify that this correspondence is being deposited with the United States Postal Service as Express Mail with sufficient postage on the date indicated above and is addressed to the Commissioner for Patents, U.S. Patent and Trademark Office, P.O. Box 2327, Arlington, VA 22202.

Signature

Gabe Lewis
Typed or Printed Name of Person Signing Certificate

UNIFIED DESIGN PARAMETER DEPENDENCY
MANAGEMENT METHOD AND APPARATUS

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims priority from U.S. Provisional
Application No. 60/315,852, filed August 29, 2001, and titled
"Visual Modeling and Design Capture Environment," which is
5 incorporated by reference.

TECHNICAL FIELD

This invention relates to a unified design parameter
dependency management method and apparatus.

BACKGROUND

10 The logic design process may be broken into different
tasks, such as, for example, design, capture, documentation,
compilation, simulation, and debug. Many logic designers may be
involved in the logic design process. During some of the tasks,
such as, for example, design and capture, managing design
15 parameters (e.g., signal parameters) may be a complex and time-
consuming process, especially when design changes are made that
may affect numerous aspects of the logic design.

DESCRIPTION OF DRAWINGS

Fig. 1 is a block diagram of a computer system.

Fig. 2 is a block diagram of a data word.

Fig. 3 is a block diagram of a table in a database.

Fig. 4A is a block diagram of a portion of computer code.

Fig. 4B is a block diagram of a portion of computer code.

5 Fig. 5 is a flow chart of a process for managing signal parameters.

Like reference symbols in the various drawings indicate like elements.

DETAILED DESCRIPTION

10 Fig. 1 illustrates an exemplary system 100 such as a computer system that may be used in the logic design process. The system 100 may include various input/output (I/O) devices (e.g., mouse 103, keyboard 105, and display 107) and a general purpose computer 110 having central processor unit (CPU) 120,
15 I/O unit 130, memory 140, and storage 150. Storage 150 may store machine-executable instructions, data, and various programs such as an operating system 152, one or more application programs 154, and one or more databases 156, all of which may be processed by CPU 120.

20 System 100 also may include a communications card or device 160 (e.g., a modem and/or a network adapter) for exchanging data with a network 170 using a communications link 175 (e.g., a telephone line, a wireless network link, a wired network link,

or a cable network). Other examples of system 100 may include a personal computer, a desktop computer, a notebook computer, a handheld device, a workstation, a server, a device, a component, other equipment, or some combination of these capable of responding to and executing instructions in a defined manner.

System 100 may be arranged to operate in concert with one or more similar systems to facilitate the logic design process. At least one of the systems 100 may include a central database, such as, for example, database 156 stored in storage 150, for use during the logic design process. The central database may be stored on one system 100 or may be partitioned and stored on more than one system 100. The central database typically is accessible to the users involved in the logic design process (e.g., designers, architects). For example, the central database may be integrated into the logic design process as part of a logic design module that performs one or more of the logic design tasks.

In a logic design, wires or signals may be used to connect the various components (e.g., logic blocks, logic gates). The signals may include a single bit signal or a multiple bit signal. The signals may be represented in a graphical model, which also may include a textual description, and/or a textual model.

In the case of multiple bit signals, the bit width of the signal typically must match the width of the pins on the logic component to which the signal connects. A multiple bit signal may be defined by one or more signal parameters. The signal parameters may include a signal label, a signal width, one or more bit fields with labels for each bit field, the width of the bit fields, and the position of the bit fields within the signal.

Multiple signals with varying signal parameters may be used in a logic design. For example, Fig. 2 illustrates a multiple-bit signal 200 that includes a three bit parity field 210 located in bit positions 0-2, a three bit status field 220 located in bit positions 3-5, a ten bit lower data field 230 located in bit positions 6-15, and a sixteen bit upper data field 240 located in bit positions 16-31. Signal 200 may be used by numerous designers in the logic design process and may permeate throughout the logic design and interface with multiple components.

The central database, described above, may be used to manage the numerous signals and signal parameters. The central database provides a single repository to store the signal parameters used in the logic design. The central database may be searchable.

The signal parameters may be arranged and stored in the central database in various formats. For example, Fig. 3 illustrates a table 300 using one possible format to define the signal parameters related to signal 200 of Fig. 2. Table 300 includes a label for the signal (Data_Word_Range) and the bit positions for the signal (31:0). Table 300 also includes a label for the bit fields within the signal and their corresponding bit positions (e.g., Upper_Data_Range with bit positions (31:16), Lower_Data_Range with bit positions (15:6), Status_Range with bit positions (5:3), and Parity_Range with bit positions (2:0)). Table 300 also includes a separate label for the bit width of the signal and each of the bit fields (e.g., Data_Word_Width is 32 bits wide, Upper_Data_Width is 16 bits wide, Lower_Data_Width is 10 bits wide, Status_Width is 3 bits wide, and Parity_Width is 3 bits wide).

Logic designers and architects use the signal parameters maintained in the central database when performing the tasks of logic design (e.g., design and capture). The central database is integrated with the logic design tool used by the logic designers and architects in performing these logic design tasks. For example, the logic design tool may be used to create textual format and/or graphical format logic designs. When two different designers use the same signal parameters in a portion of the design, uniformity of design is possible. Additionally,

when modifications are made to the signals and the signal parameters, the modifications may be made in the central database without necessarily having to change the logic already created by each individual designer.

5 For example, Fig. 4A illustrates a table 400 of a portion of computer code that may be used as part of a logic design. The computer code may include a Hardware Design Language (HDL), such as Verilog (IEEE Standard 1364) or a Very High Speed Integrated Circuit Hardware Design Language (VHDL). Verilog
10 includes a textual format for describing a logic design that includes electronic circuits and systems.

 The computer code uses the signal parameter labels defined in the central database to construct the logic design. The logic design tool that may be used to create the computer code
15 in table 400 is linked to the integrated central database that includes the parameter labels and their definitions. Additionally, the central database is compatible with the format of the computer code. For instance, the central database is compatible with computer code that includes a "TIC_define"
20 format (e.g., used with Verilog) for use with signal parameters.

 Thus, if a change is made to one or more of the signal parameters, the change is made in the database and the computer code is updated automatically to reflect the changes made in the database. Because the labels are used in the code, the code

itself does not need to be rewritten to reflect modifications to the signal parameters.

In contrast, for example, Fig. 4B illustrates a table 450 of a portion of computer code that uses constants, instead of the signal parameter labels defined in the central database, to construct the logic design. In this instance, since the signal parameters labels were not used, there is no relationship between the logic design and the central database. When a signal parameter is modified, the computer code that includes the constants needs to be manually updated. Manually updating the computer code may be a complex and time-consuming task.

Fig. 5 illustrates a process 500 for managing signal parameters. Process 500 typically includes defining one or more signal parameters with a value (510), maintaining the defined signal parameters in a central database (520), and using the defined signal parameters that are maintained in the central database in computer code for a logic design (530).

Process 500 may further include modifying the value of the defined signal parameter in the central database, and automatically modifying the logic design with the modified value of the defined signal parameter. Thus, the computer code does not need to be modified manually when a signal parameter modification is made.

When a signal parameter is modified in the database, the textual format and/or the graphical format of the logic design may indicate any discrepancies that may have resulted from the signal parameter modification. For example, if the width of a signal is modified in the central database, then a discrepancy between the signal and any of the logic components that use that signal may be indicated. For instance, if the signal was originally defined in the central database as having a width of thirty-two bits and the width is subsequently modified to a width a sixty-four bits, then any occurrence of that signal in the logic design may be flagged for a user to see that a change to the signal has been made.

The indication that a discrepancy exists may be a visual indication in the graphic format of the logic design, such as, by changing the color of the signal (e.g., changing the color of the signal from green to red, or changing the color of a logic component or a portion of the logic component to which the signal connects from green to red), or by bolding, highlighting, italicizing, or providing some other visual cue as to the discrepancy. A similar visual indication may be made in the textual format of the logic design (e.g., a visual indication in the computer code for the logic design).

Additionally or alternatively, a textual cue may be provided in the graphic format of the logic design to indicate that a discrepancy exist.

Other examples of signal parameter modifications that may be made include modifying the position of a bit field within a signal, modifying the width of a bit field within the signal, and modifying the width of the signal. Modifications may include additions, deletions, and updates.

The described systems, methods, and techniques may be implemented in digital electronic circuitry, computer hardware, firmware, software, or in combinations of these elements. Apparatus embodying these techniques may include appropriate input and output devices, a computer processor, and a computer program product tangibly embodied in a machine-readable storage device for execution by a programmable processor.

A process embodying these techniques may be performed by a programmable processor executing a program of instructions to perform desired functions by operating on input data and generating appropriate output. The techniques may be implemented in one or more computer programs that are executable on a programmable system including at least one programmable processor coupled to receive data and instructions from, and to transmit data and instructions to, a data storage system, at least one input device, and at least one output device. Each

computer program may be implemented in a high-level procedural or object-oriented programming language, or in assembly or machine language if desired; and in any case, the language may be a compiled or interpreted language. Suitable processors include, by way of example, both general and special purpose microprocessors. Generally, a processor will receive instructions and data from a read-only memory and/or a random access memory. Storage devices suitable for tangibly embodying computer program instructions and data include all forms of non-volatile memory, including by way of example semiconductor memory devices, such as Erasable Programmable Read-Only Memory (EPROM), Electrically Erasable Programmable Read-Only Memory (EEPROM), and flash memory devices; magnetic disks such as internal hard disks and removable disks; magneto-optical disks; and Compact Disc Read-Only Memory (CD-ROM). Any of the foregoing may be supplemented by, or incorporated in, specially-designed ASICs (application-specific integrated circuits).

It will be understood that various modifications may be made. For example, advantageous results still could be achieved if steps of the disclosed techniques were performed in a different order and/or if components in the disclosed systems were combined in a different manner and/or replaced or supplemented by other components. Also, for instance, it is possible to use a central database that includes design

parameters or a portion of that central database for more than one logic design. Accordingly, other implementations are within the scope of the following claims.